Università degli Studi di Parma
Dipartimento di Ingegneria e Architettura

Sistemi operativi e in tempo reale – a.a. 2022/23

Analisi di schedulabilità

Analisi di schedulabilità

- Approcci:

- Utilizzazione schedulabile dell'algoritmo
  - con EDF Utot≤1; con RM: LL, KM, HB, altri
- Analisi del tempo di risposta ovvero del tempo di processore richiesto:
  - Metodo strutturato per effettuare analisi da istante critico o con relazioni di fase definite
  - Anche in varianti algebriche / computazionali
- Altri metodi:
  - simulazione, dry run, hardware-in-the-loop, finger-crossed method

Analisi del tempo di risposta (Audsley et al., '90)

- Denominata anche *time-demand analysis*, realizza un *test di schedulabilità* per *task con priorità fissa*
- Si applica ad *algoritmi a priorità fissa arbitrari*; di particolare utilità quando manca una stima ragionevole della utilizzazione schedulabile (ad es., DM)
- Si basa sull'idea di generare la situazione di caso peggiore, l'*istante critico*, e di simulare gli eventi fino alla prima deadline di ogni task
- Dati di ingresso: $\{\tau_i\} = \{(T_i, C_i, D_i)\}$
- L'algoritmo di Audsley verifica un task alla volta per determinare se i *tempi di risposta* dei job, calcolati a partire da (T_i, C_i) rispettano le deadline relative D_i

Analisi del tempo di risposta

- Ipotesi: $D_i \leq T_i$, $U \leq 1$

  (per estensione al caso di deadline relative arbitrarie, v. Liu)
- Osservazione: ogni task è influenzato solo dai task a più alta priorità, da cui subisce revoche
- Si inizia dal task a priorità più elevata e si procede per priorità decrescente
- R_i = *massimo tempo di risposta* di τ_i; si ottiene a partire dall'istante critico t_0. τ_i è schedulabile se:

$$R_i = C_i + I_i \leq D_i \quad i=1,2,..,N$$

- I_i è l'*interferenza* sul tempo di risposta di τ_i da parte di job a più alta priorità dovuta a preemption

Analisi del tempo di risposta

- In generale, in un istante t_0+t la *richiesta totale di tempo di processore* (time-demand) $W_i(t)$ di un job di τ_i e di tutti gli altri job a più alta priorità rilasciati in $[t_0, t_0+t]$ è:

$$W_i(t) = C_i + \sum_{k=1,i-1} \lceil t/T_k \rceil C_k \qquad \text{per } 0 < t \leq T_i$$

- Il job di τ_i può rispettare la sua deadline t_0+D_i se in un qualche istante t_0+t entro la deadline la *disponibilità* di tempo di processore, t, soddisfa la *richiesta totale* $W_i(t)$:

$$\exists\ t \mid W_i(t) \leq t \qquad \text{per qualche } t \leq D_i$$

Analisi del tempo di risposta

- Se $W_i(t) > t$ per ogni $t \leq D_i$

  il job *non può* completare entro la deadline, nel caso venga rilasciato nell'istante critico
- L'insieme di task non è garantito in base al test di Audsley
- L'insieme di task *potrebbe* essere garantito solo se è possibile controllare la fase di rilascio ed il jitter dei task in modo da evitare situazioni di rilascio all'istante critico
- Se il test di Audsley non è soddisfatto, per garantire i task occorre simularne l'evoluzione, con le esatte fasi di rilascio previste, per un iperperiodo (assumendo assenza di jitter)

Time-demand analysis

- La tecnica è denominata *time-demand analysis*, analisi della funzione di *richiesta del tempo di processore*

- La funzione di time-demand $W_i(t)$ del task τ_i è:

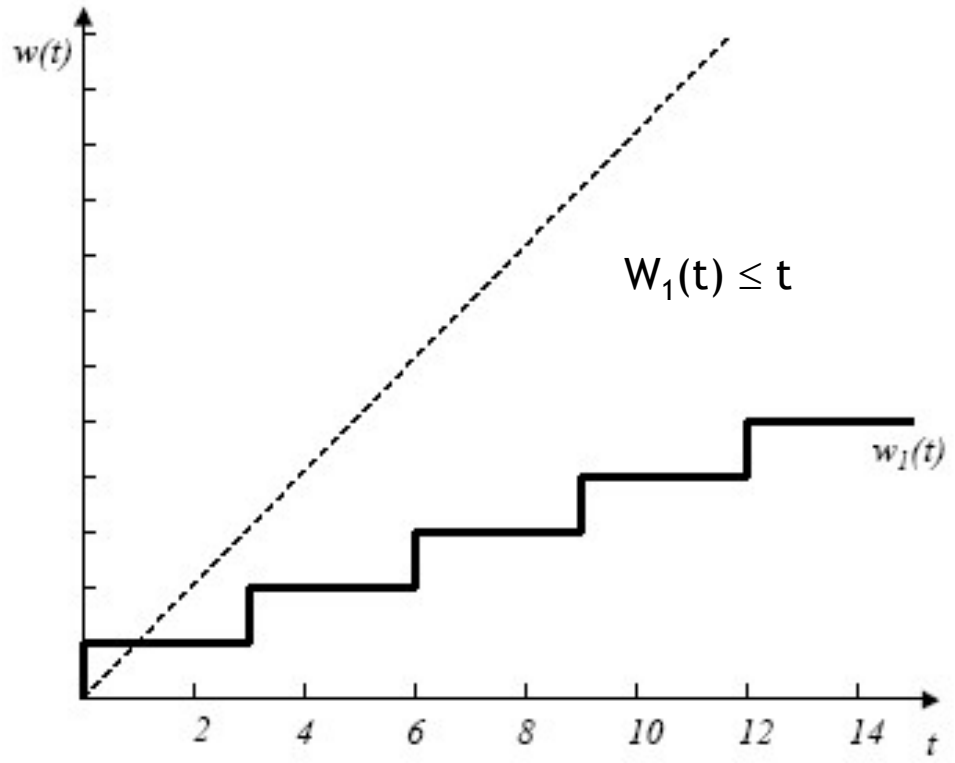
$$W_i(t) = C_i + \sum_{k=1,i-1} \lceil t/T_k \rceil C_k \qquad \text{per } 0 < t \leq T_i$$

- Per qualunque assegnazione statica di priorità, per task periodici con deadline $D_i \leq T_i$ è possibile eseguire l'analisi del tempo richiesto per via grafica, a partire dall'istante critico

Analisi del tempo di risposta

- Grafico del tempo richiesto al processore dal task τ_i:

Ad es., per τ_1 =(3,1), a massima priorità:



L'ordinata rappresenta la *richiesta* di processore, la retta a pendenza 1 la *disponibilità* di tempo

Analisi del tempo di risposta - Esempio

τ_1 =(3,1) U_1=0.333

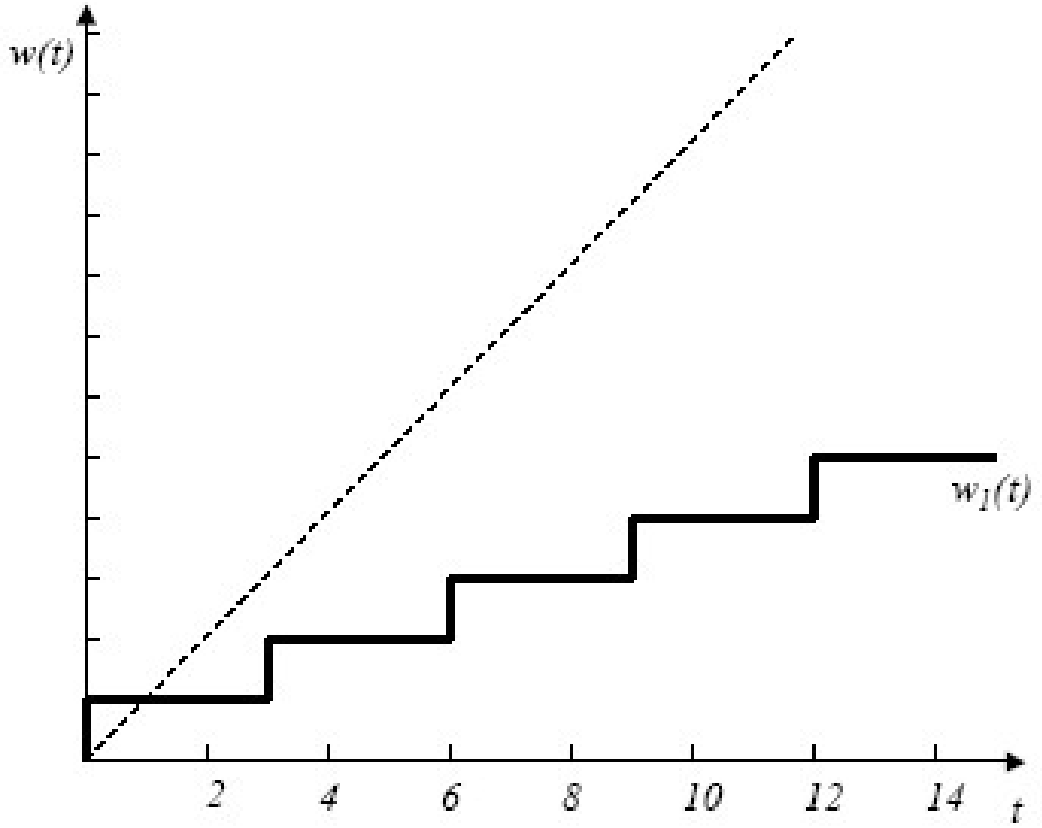
τ_2 =(5,1.5) U_{12}=0.633

τ_3 =(7,1.25) U_{123}=0.812

τ_4 =(8,0.5) U_{1234}=0.874

In base a U_{LL}(RM) e HB, solo τ_1 e τ_2 sono garantiti

τ_1 completa in t=1 e rispetta la deadline (è a max priorità)



Analisi del tempo di risposta - Esempio

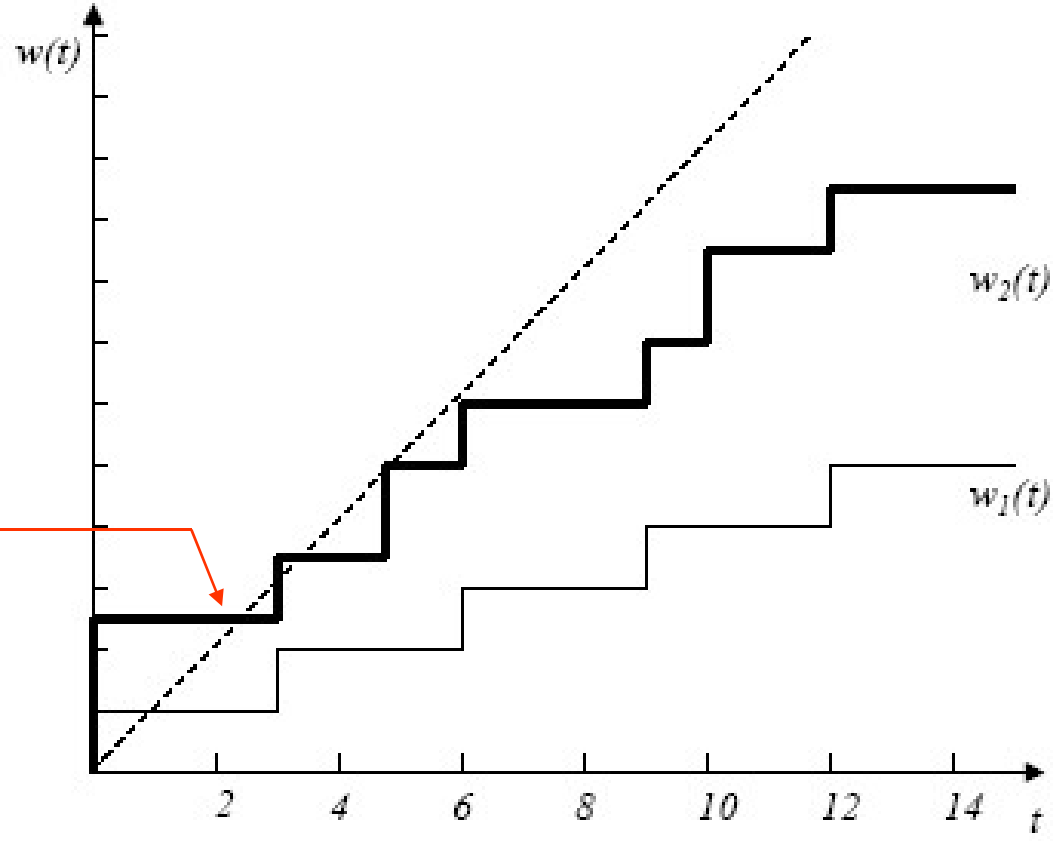
τ_1 =(3,1)

τ_2 =(5,1.5)

τ_3 =(7,1.25)

τ_4 =(8,0.5)

- All'istante 2.5 la richiesta totale di τ_2 è W_2=2.5 ≤ D_2=5
- τ_2 rispetta la deadline



Analisi del tempo di risposta - Esempio

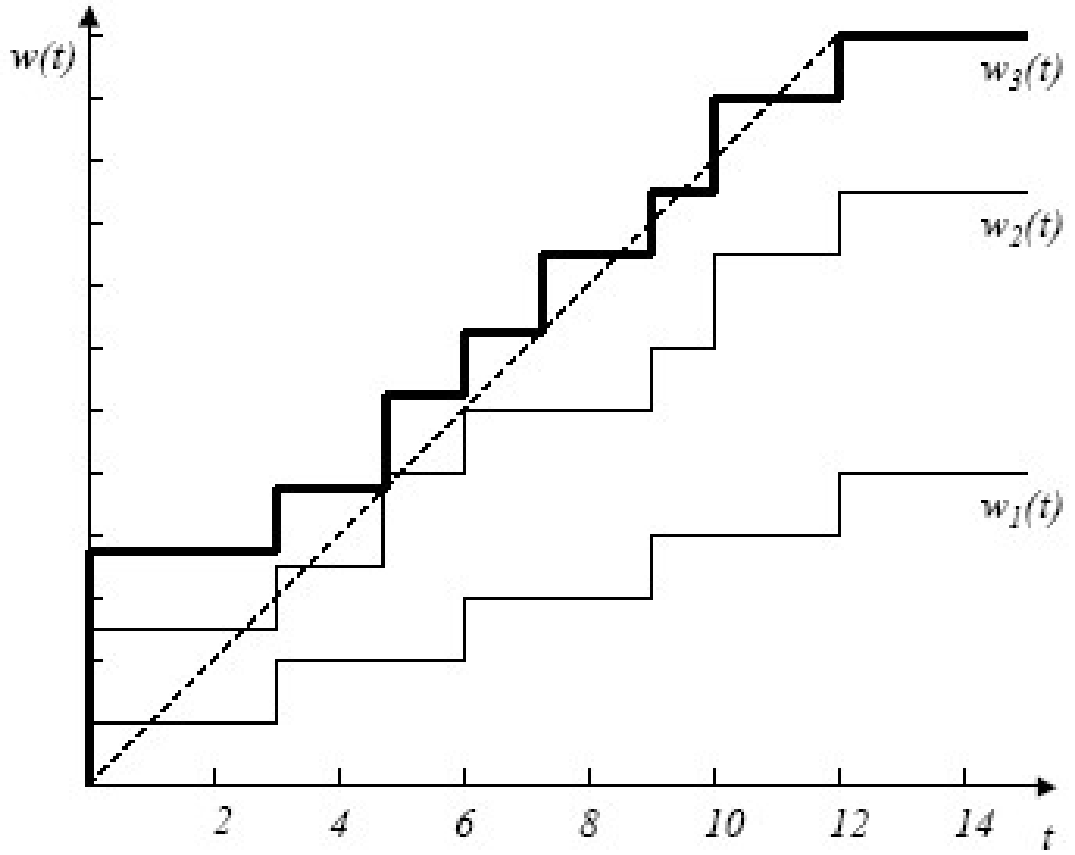
τ_1 =(3,1)

τ_2 =(5,1.5)

τ_3 =(7,1.25)

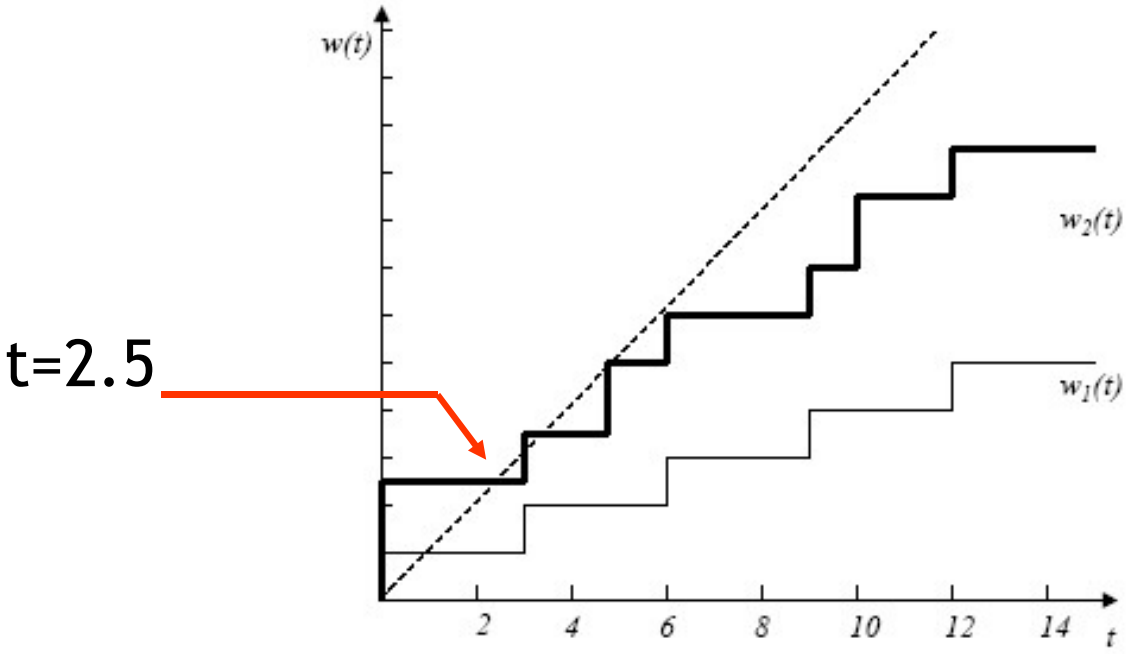
τ_4 =(8,0.5)

- All'istante 4.75 la richiesta totale di τ_3 è W_3=4.75 ≤ D_3=7
- τ_3 rispetta la deadline



Analisi del tempo di risposta - Esempio

- Il punto di intersezione tra $W_i(t)$ e la retta y=t individua l'istante in cui τ_i completa a partire dal suo istante critico



Analisi del tempo di risposta - Esempio

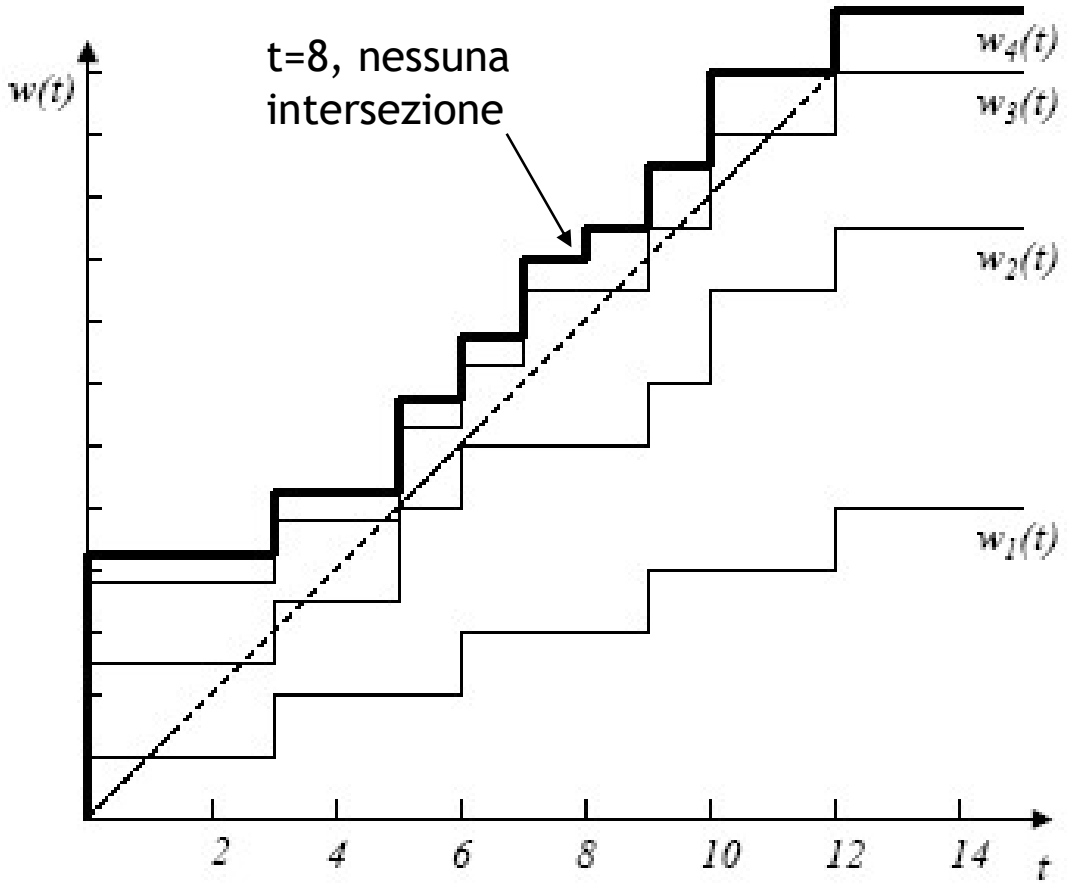
- nessuna intersezione tra W_4 e y=t entro D_4
- τ_4 non rispetta la deadline

τ_1 =(3,1)

τ_2 =(5,1.5)

τ_3 =(7,1.25)

τ_4 =(8,0.5)



Analisi del tempo di risposta - Esempio

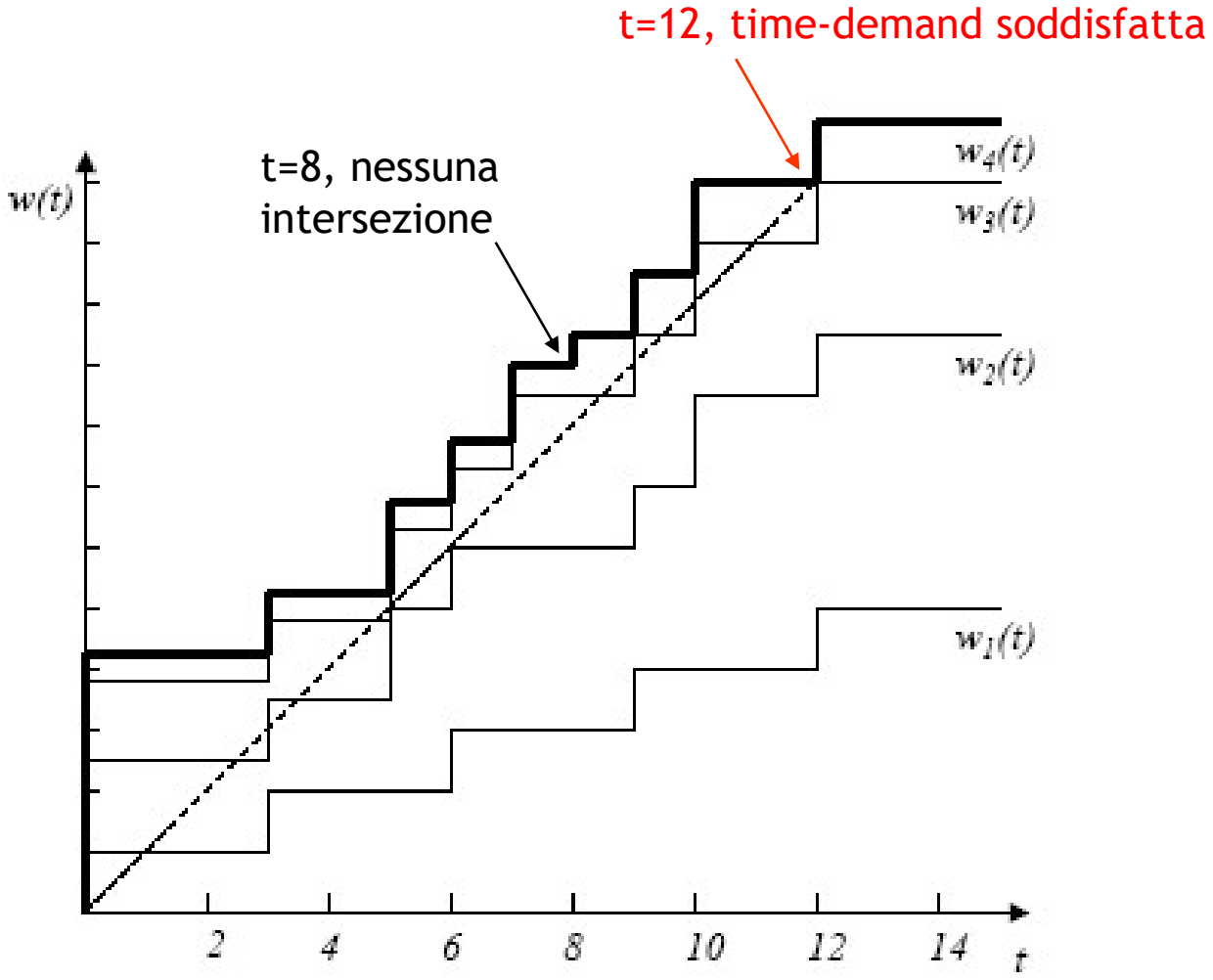
- nessuna intersezione tra W_4 e y=t entro D_4
- τ_4 non rispetta la deadline
- $W_4(t)$ tocca y=t in t=12, quindi entro 1.5 periodi recupera e si mette in pari
- backlog max di τ_4 =1

τ_1 =(3,1)

τ_2 =(5,1.5)

τ_3 =(7,1.25)

τ_4 =(8,0.5)



Analisi del tempo di risposta

Task set modificato:

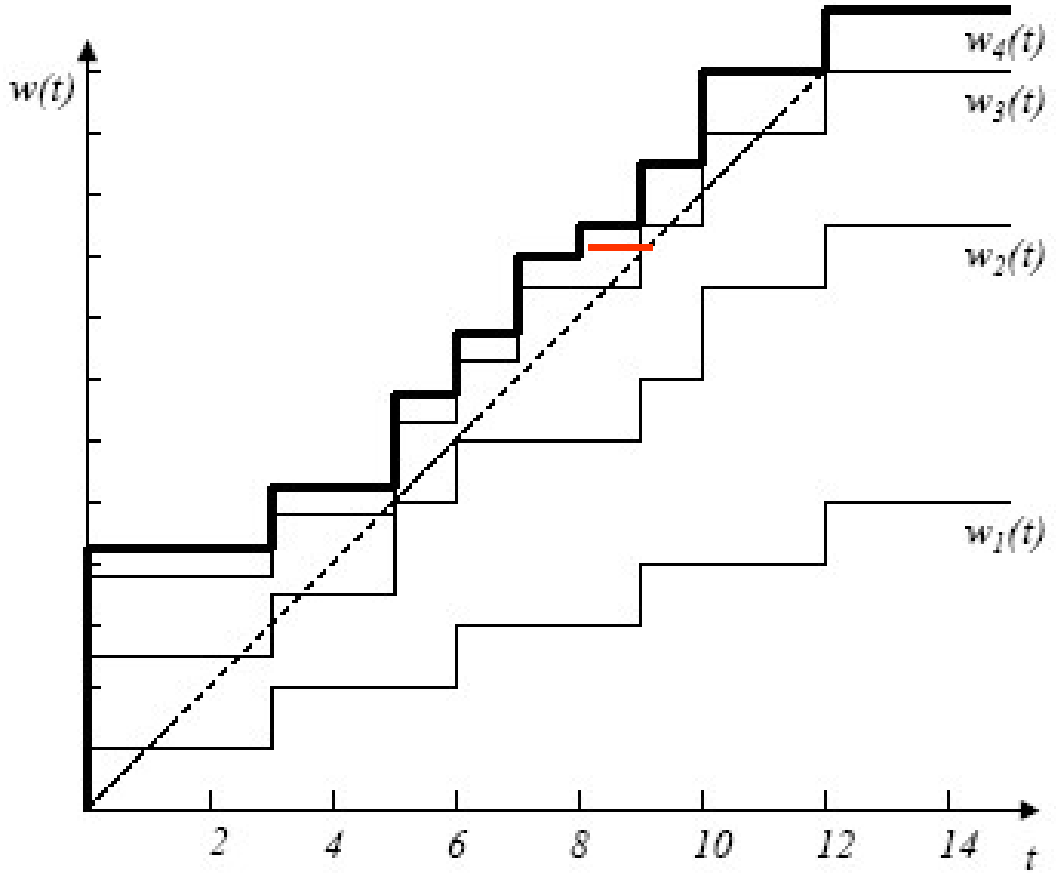
τ_1 =(3,1)

τ_2 =(5,1.5)

τ_3 =(7,1.25)

τ_4 =(9,0.5)

all'istante t=9 la richiesta totale di τ_4 è W_4=9 e la deadline è soddisfatta



Analisi del tempo di risposta

- Esercizio: Modificare i parametri di $\tau 4$ nell'esempio precedente
  - $\tau 4=(10,1)$ -- ce la fa?
  - $\tau 4=(12,1)$ -- ce la fa?

Analisi del tempo di risposta

- Osservazioni:
- La funzione di time-demand di ogni task è una *scalinata*
- La funzione sale negli *istanti multipli interi* dei periodi dei task a più alta priorità e del task stesso
- Immediatamente prima della salita di un gradino, la differenza tra tempo di processore richiesto e tempo disponibile, $W_i(t) - t$, è al valore minimo dall'alzata precedente
- Se non interessa calcolare il massimo tempo di risposta ma *solo se il task è schedulabile*, è sufficiente verificare se $W_i(t) \leq t$ in questi istanti

Algoritmo di analisi del tempo di risposta

- Calcoliamo $W_i(t) = C_i + \sum_{k=1,i-1} \lceil t/T_k \rceil C_k$
- Verifichiamo se $W_i(t) \leq t$ per i valori $t=jT_k$, con $k=1,2,..i$, e $j=1,2,... \lfloor \min(T_i, D_i)/T_k \rfloor$

k seleziona i task a priorità maggiore di τi
j identifica i rilasci che interferiscono nell'intervallo critico

- Se la disuguaglianza è soddisfatta in almeno un istante, τ_i è schedulabile

- Costo: $O(n\, q_{n,1})$, ove $q_{n,1} = T_n/T_1$

Algoritmo di analisi del tempo di risposta

- Nell'esempio: τ_1=(3,1), τ_2=(5,1.5), τ_3=(7,1.25), τ_4=(9,0.5)
- τ_1 e τ_2 sono garantiti
- Per τ_3 dobbiamo verificare $W_3(t) \leq t$ negli istanti:
  k=1: $\lfloor T3/T1 \rfloor = \lfloor 7/3 \rfloor = 2$ → t=3, t=6
  k=2: $\lfloor T3/T2 \rfloor = \lfloor 7/5 \rfloor = 1$ → t=5
  k=3: $\lfloor T3/T3 \rfloor = 1$ → t=7
- L'insieme degli istanti in cui esaminare $W_3(t)$ è (3,5,6,7)
- Si verifica che $W_3(t) \leq t$ non è soddisfatta per t=3 ma è soddisfatta per t=5
- Il massimo tempo di risposta di τ_3 è compreso tra 3 e 5. Poiché $R_3 \leq 5$, $R_3 \leq D_3$ e τ_3 è garantito

Analisi del tempo di risposta

- Il metodo *è robusto rispetto a variazioni*, anche transitorie, dei parametri dei task:
- Se il *tempo di esecuzione* di qualche job di τ_i *è inferiore al WCET* considerato per l'analisi, la curva $W_j(t)$ di tutti i job a priorità inferiore a τ_i si abbassa, e quindi aumenta la possibilità di soddisfare il vincolo
- Se *l'intervallo tra due rilasci* di qualche job di τ_i *è superiore al periodo* indicato, la curva $W_j(t)$ dei job a priorità inferiore sale più lentamente

Calcolo del tempo di risposta

- L'analisi del tempo di risposta consente di aggiungere, sulle curve di time-demand, intervalli di non revocabilità dei job e situazioni di autosospensione
- → consente analisi di schedulabilità sotto ipotesi più generali

- Nel caso più semplice, è possibile calcolare il tempo di risposta tramite un algoritmo iterativo (Audsley)

Calcolo del tempo di risposta

- Algoritmo iterativo:
- Si inizia con una stima pari al tempo di esecuzione C_i

$$\begin{cases} R_i^0 = C_i \\ R_i^s = C_i + \sum_{k=1}^{i-1} \left\lceil \frac{R_i^{(s-1)}}{T_k} \right\rceil C_k \end{cases}$$

- Si itera fino a quando non si ha $R_i^s = R_i^{s-1}$ oppure $R_i^s > D_i$
  - Se $R_i^s = R_i^{s-1}$ e $R_i^s \leq D_i$ il task i è garantito
  - Se $R_i^s > D_i$ il task non è garantito

Calcolo del tempo di risposta - Esempio

- $\tau_1=(3,1)$, $\tau_2=(5,1.5)$, $\tau_3=(7,1.25)$, $\tau_4=(9,0.5)$
- Per τ_3, sostituendo in $W_i(t) = C_i + \sum_{k=1,i-1} \lceil t/T_k \rceil C_k$ si ha:

$$R_3^* = 1.25+1 \lceil R_3/3 \rceil + 1.5 \lceil R_3/5 \rceil$$

- $R_3^0=1.25$
- $R_3^1=1.25+ 1 \lceil 1.25/3 \rceil + 1.5 \lceil 1.25/5 \rceil =1.25+1+1.5=3.75$
- $R_3^2=1.25+ 1 \lceil 3.75/3 \rceil + 1.5 \lceil 3.75/5 \rceil =1.25+2+1.5=4.75$
- $R_3^3=1.25+ 1 \lceil 4.75/3 \rceil + 1.5 \lceil 4.75/5 \rceil =1.25+2+1.5=4.75$
- $\rightarrow R_3=4.75 \leq D_3$, τ_3 garantito

Analisi del tempo di risposta

- Esercizi: applicare i diversi metodi nei seguenti casi
  - τ_1 =(3,1), τ_2 =(5,1.5), τ_3 =(7,1.25), τ_4 =(8,0.5)
  - τ_1 =(3,1), τ_2 =(5,1.5), τ_3 =(7,1.25), τ_4 =(9,0.5)
  - τ_1 =(3,1), τ_2 =(5,1.5), τ_3 =(7,1.25), τ_4 =(10,1)
  - τ_1 =(3,1), τ_2 =(5,1.5), τ_3 =(7,1.25), τ_4 =(12,1)

Sezioni di codice non revocabili

- Alcuni job possono essere non revocabili o avere porzioni non revocabili
- Definizione: *porzione non revocabile*

  ρ_i= la più lunga sezione non revocabile dei job di τ_i
- Definizione: Un *job* si dice *bloccato* se subisce una *inversione di priorità*, ovvero se è ritardato nell'esecuzione da un job *a priorità inferiore*
- Per verificare la schedulabilità di un task τ_i occorre considerare:
  - Tutti i task a più alta priorità, *e*
  - Le porzioni non revocabili dei task a più bassa priorità

Analisi di schedulabilità con porzioni non revocabili

- Definizione: il *tempo di blocco* B_i del task τ_i è il più lungo intervallo di tempo per cui un job di τ_i può essere bloccato da job a priorità inferiore.
- Per algoritmi a priorità statica, ordinando i task per priorità decrescente vale:

$$B_i = \max_{i+1 \le k \le n} \rho_k$$

- Funzione di time-demand in presenza di blocco:

$$W_i(t) = C_i + B_i + \sum_{k=1,i-1} \lceil t/T_k \rceil C_k \qquad \text{per } 0 < t \le \min(D_i, T_i)$$

- (Per algoritmi a priorità dinamica -> teorema di Baker, dopo)

Analisi del tempo di risposta con porzioni non revocabili

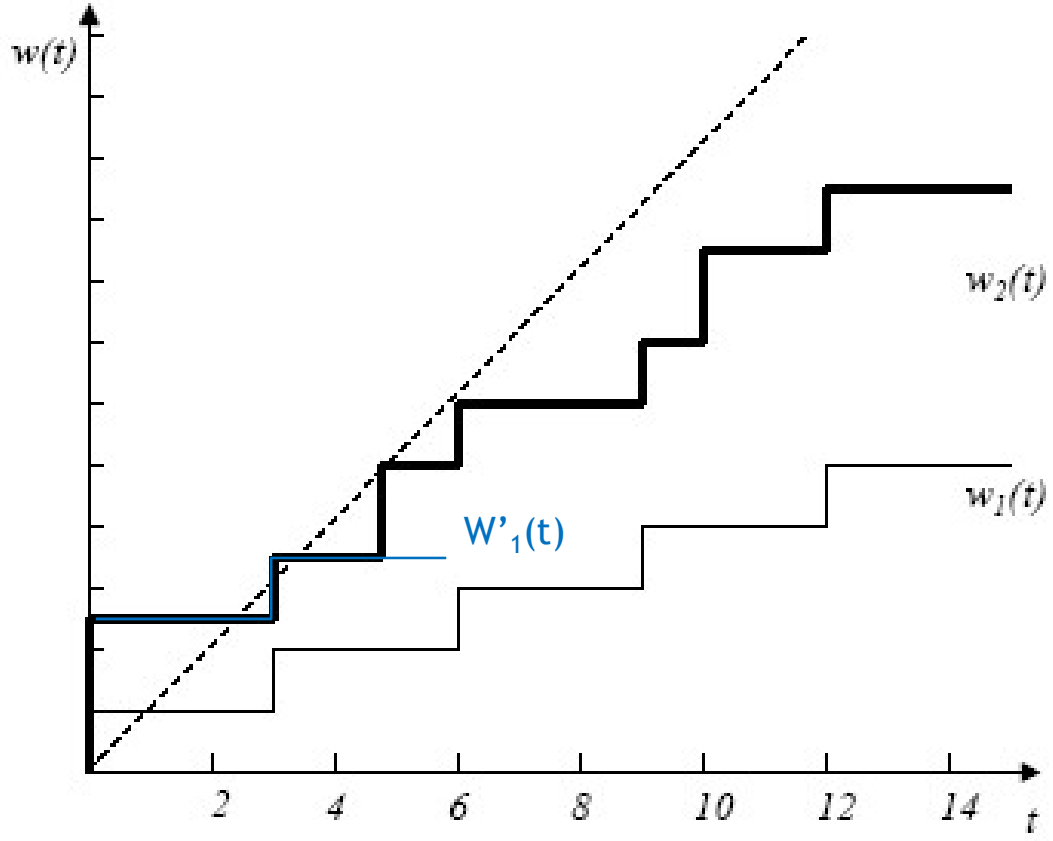
τ_1 =(3,1)

τ_2 =(5,1.5) non revoc.

τ_3 =(7,1.25)

τ_4 =(9,0.5)

- La time-demand totale di τ_2 non cambia
- La time-demand totale di τ_1 w_1'(t) aumenta di B_1=1.5



Analisi del tempo di risposta con porzioni non revocabili

- In questo caso particolare la time demand cambia solo per τ_1. Le time demand di τ_2, τ_3, τ_4 invece non cambiano
- Si osserva che τ_1 è comunque garantito, poiché completa in t=2.5 nel caso peggiore, così come accade a τ_2. Per i rimanenti task non cambia nulla: semplicemente i primi due task possono scambiarsi di priorità
- Se invece fosse τ_3 a non essere revocabile …

Bound di utilizzazione in presenza di porzioni non revocabili

- Per sistemi a priorità fissa
- Occorre considerare il tempo di blocco che può subire ogni task
- Per il task τ_i :

$$C1/T1+C2/T2+ \ldots +(C_i+B_i)/T_i = \sum_{k=1,i} C_k/T_k+B_i/T_i \leq U_x(i)$$

- Ove ad esempio x=LL o x=KM per RM
- Ogni task può subire blocchi di durata diversa e da task diversi, pertanto il test deve essere effettuato *un task alla volta* con priorità decrescente

Scheduling deadline-driven con porzioni non revocabili

- Per sistemi a priorità dinamica (EDF)
- Teorema: In un insieme di task schedulati con EDF, un job J_k con deadline relativa D_k può bloccare un job J_i con deadline relativa D_i solo se $D_k > D_i$ [Baker, 1991]

- Indicizzando i task in base alle deadline relative, il tempo di blocco di ciascun task periodico dovuto a porzioni non revocabili è ancora: $B_i = \max_{i+1 \le k \le n} \rho_k$

  ove i task sono ordinati per deadline relative crescenti

Scheduling deadline-driven con porzioni non revocabili

- Dimostrazione [Baker, 1991]:

- J_k può bloccare J_i solo se ha priorità inferiore. Poiché le priorità sono assegnate in modo EDF questo significa che J_k ha una deadline posteriore, ovvero $d_k > d_i$
- Inoltre, quando il job a priorità maggiore J_i è stato rilasciato J_k doveva già essere in esecuzione per bloccare J_i , Questo significa che J_k è stato rilasciato prima, ovvero $r_k < r_i$
- Le due disuguaglianze sono soddisfatte solo se $D_k > D_i$ □

Scheduling deadline-driven con porzioni non revocabili

- *Un task* τ_i con un tempo di blocco B_i è schedulabile con altri task periodici indipendenti dall'algoritmo EDF su un processore se

$$\sum_{k=1,n} C_k/\min(D_k,T_k) + B_i/\min(D_i,T_i) \leq 1$$

- *Il sistema* è schedulabile se la condizione è verificata per ogni i=1, 2, …, n

Scheduling deadline-driven con porzioni non revocabili

- *Un task* τ_i con un tempo di blocco B_i è schedulabile con altri task periodici indipendenti dall'algoritmo EDF su un processore se

$$\sum_{k=1,n} C_k/\min(D_k,T_k) + B_i/\min(D_i,T_i) \leq 1$$

risultato negativo per EDF: la condizione considera tutti i task !

- *Il sistema* è schedulabile se la condizione è verificata per ogni i=1, 2, …, n

Es.: τ_1=(3,1), τ_2=(5,1.5), τ_3=(7,1.25), τ_4=(9,0.5);
τ_3 non revocabile in sezione di durata=1